SQLite数据库2

主讲：万永权

主要内容

Contents

1. API建库、表
2. 数据操作：增删改查

8.3 数据库存储

8.3.3 代码建库

- ✓ 在代码中动态建立数据库是比较常用的方法
- ✓ 例如在程序运行过程中，当需要进行数据库操作时，应用程序会首先尝试打开数据库，此时如果数据库并不存在，程序则会自动建立数据库，然后再打开数据库
- ✓ 在编程实现时，一般将所用对数据库的操作都封装在一个类中，因此只要调用这个类，就可以完成对数据库的添加、更新、删除和查询等操作

SQLite相关类

- SQLiteOpenHelper：抽象类，我们通过继承该类，然后重写数据库创建以及更新的方法， 我们还可以通过该类的对象获得数据库实例，或者关闭数据库！
- SQLiteDatabase：数据库访问类：我们可以通过该类的对象来对数据库做一些增删改查的操作
- Cursor：游标，有点类似于JDBC里的resultset，结果集！可以简单理解为指向数据库中某 一个记录的指针！

SQLiteOpenHelper

使用SQLiteOpenHelper类创建数据库与版本管理

- 对于涉及数据库的app,我们不可能手动地去创建数据库文件,所以需要在第一次启用app 的时候就创建好数据库表;而当我们的应用进行升级需要修改数据库表的结构时,这个时候就需要 对数据库表进行更新了;对于这两个操作,安卓给我们提供了SQLiteOpenHelper的两个方法, onCreate( )与onUpgrade( )来实现

SQLite使用

1、SQLiteOpenHelper 具体方法：

onCreate(SQLiteDatabase db)	一个SQLiteDatabase对象作为参数，当数据库第一次被建立的时候被执行，例如创建表，初始化数据等。
onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)	需要三个参数，一个SQLiteDatabase对象，一个旧的数据库版本号（oldVersion）和一个新的数据库版本号（newVersion），当数据库需要被更新的时候执行，例如删除久表，创建新表，这样就可以把一个数据库从旧的模型转变到新的模型。
getReadableDatabase()	得到可读的数据库，返回SQLiteDatabase对象，然后通过对象进行数据库读取操作。
getWritableDatabase()	得到可写的数据库，返回SQLiteDatabase对象，然后通过对象进行数据库写入或者读取操作。
onOpen(SQLiteDatabase db)	当打开数据库时的回调函数。
close()	关闭数据库，需要强调的是，在每次打开数据库之后停止使用时调用，否则会造成数据泄露。

8.3 数据库存储

8.3.3 代码建库

- 下面内容是DBAdapter类的部分代码，封装了数据库的建立、打开和关闭等操作

```
1   public class DBAdapter {
2    private static final String DB_NAME = "people.db";
3    private static final String DB_TABLE = "peopleinfo";
4    private static final int DB_VERSION = 1;
5
6    public static final String KEY_ID = "_id";
7    public static final String KEY_NAME = "name";
8    public static final String KEY_AGE = "age";
9    public static final String KEY_HEIGHT = "height";
10
11   private SQLiteDatabase db;
```

8.3 数据库存储

8.3.3 代码建库

```
12    private final Context context;
13    private DBOpenHelper dbOpenHelper;
14
15    private static class DBOpenHelper extends SQLiteOpenHelper {}
16
17    public DBAdapter(Context _context) {
18          context = _context;
19    }
20
21    public void open() throws SQLiteException {
22          dbOpenHelper = new DBOpenHelper(context, DB_NAME, null,
      DB_VERSION);
23          try {
24    db = dbOpenHelper.getWritableDatabase();
```

8.3 数据库存储

8.3.3 代码建库

```
25          }catch (SQLiteException ex) {
26      db = dbOpenHelper.getReadableDatabase();
27          }
28      }
29
30      public void close() {
31          if (db != null){
32      db.close();
33      db = null;
34          }
35      }
36      }
```

8.3 数据库存储

8.3.3 代码建库

- 从代码的第2行到第9行可以看出，在DBAdapter类中首先声明了数据库的基本信息，包括数据库的文件名称、表名称和版本号，以及数据库表的属性名称
- 从这些基本信息上不难发现，这个数据库与前一小节手动建立的数据库是完全相同的
- 代码第11行声明了SQLiteDatabase的实例
- SQLiteDatabase类封装了较多的方法，用以建立、删除数据库，执行SQL命令，对数据进行管理等工作

8.3 数据库存储

8.3.3 代码建库

- 代码第13行声明了一个非常重要的帮助类SQLiteOpenHelper，这个帮助类可以辅助建立、更新和打开数据库
- 虽然在代码第21行定义了open()函数用来打开数据库，但open()函数中并没有任何对数据库进行实际操作的代码，而是调用了SQLiteOpenHelper类的getWritableDatabase()函数和getReadableDatabase()函数
- 这个两个函数会根据数据库是否存在、版本号和是否可写等情况，决定在返回数据库实例前，是否需要建立数据库

8.3 数据库存储

8.3.3 代码建库

- ✓在代码第30行的close()函数中，调用了SQLiteDatabase实例的close()方法关闭数据库
- ✓这是代码中唯一一处直接调用了SQLiteDatabase实例的方法
- ✓SQLiteDatabase中也封装了打开数据库的函数openDatabases()和创建数据库函数openOrCreateDatabases()，因为代码中使用了帮助类SQLiteOpenHelper，从而避免直接调用SQLiteDatabase的打开和创建数据库的方法，简化了数据库打开过程中繁琐的逻辑判断过程

8.3 数据库存储

8.3.3 代码建库

DBOpenHelper继承了帮助类SQLiteOpenHelper，重载了onCreate()函数和onUpgrade()函数，代码如下：

```
1    private static class DBOpenHelper extends SQLiteOpenHelper {
2   public DBOpenHelper(Context context, String name, CursorFactory factory, int
    version){
3           super(context, name, factory, version);
4           }
5   private static final String DB_CREATE = "create table " +
6          DB_TABLE + " (" + KEY_ID + " integer primary key autoincrement, " +
7          KEY_NAME+ " text not null, " + KEY_AGE+ " integer," + KEY_HEIGHT
    + " float);";
8
9   @Override
```

8.3 数据库存储

8.3.3 代码建库

```
10  public void onCreate(SQLiteDatabase _db) {
11          _db.execSQL(DB_CREATE);
12          }
13
14  @Override
15  public void onUpgrade(SQLiteDatabase _db, int _oldVersion, int _newVersion) {
16          _db.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);
17          onCreate(_db);
18  }
19     }
```

8.3 数据库存储

8.3.3 代码建库

- 代码的第5行到第7行是创建表的SQL命令
- 代码第10行和第15行分别重载了onCreate()函数和onUpgrade()函数，这是继承SQLiteOpenHelper类必须重载的两个函数
- onCreate()函数在数据库第一次建立时被调用，一般用来创建数据库中的表，并完成初始化工作
- 在代码第11行中，通过调用SQLiteDatabase实例的execSQL()方法，执行创建表的SQL命令
- onUpgrade()函数在数据库需要升级时被调用，一般用来删除旧的数据库表，并将数据转移到新版本的数据库表中
- 在代码第16行和第17行中，为了简单起见，并没有做任何数据转移，而仅仅删除原有的表后建立新的数据表

8.3 数据库存储

8.3.3 代码建库

- ✓ 程序开发人员不应直接调用onCreate()和onUpgrade()函数，而应由SQLiteOpenHelper类来决定何时调用这两个函数
- ✓ SQLiteOpenHelper类的getWritableDatabase()函数和getReadableDatabase()函数是可以直接调用的函数
- ✓ getWritableDatabase()函数用来建立或打开可读写的数据库实例，一旦函数调用成功，数据库实例将被缓存，在需要使用数据库实例时就可以调用这个方法获取数据库实例，务必在不使用时调用close()函数关闭数据库
- ✓ 如果保存数据库文件的磁盘空间已满，调用getWritableDatabase()函数则无法获得可读写的数据库实例，这时可以调用getReadableDatabase()函数，获得一个只读的数据库实例

8.3 数据库存储

8.3.3 代码建库

- 如果不希望使用SQLiteOpenHelper类，也可以直接使用SQL命令建立数据库，方法是：
  - 先调用openOrCreateDatabases()函数创建数据库实例
  - 然后调用execSQL()函数执行SQL命令，完成数据库和数据表的建立过程，其示例代码如下：

```
1  private static final String DB_CREATE = "create table " +
2   DB_TABLE + " (" + KEY_ID + " integer primary key autoincrement, " +
3   KEY_NAME+ " text not null, " + KEY_AGE+ " integer," + KEY_HEIGHT + "
    float);";
4  public void create() {
5   db.openOrCreateDatabases(DB_NAME, context.MODE_PRIVATE, null)
6   db.execSQL(DB_CREATE);
7  }
```

8.3 数据库存储

8.3.4 数据操作

- 数据操作指的是对数据的添加、删除、查找和更新操作
- 虽然程序开发人员完全可以通过执行SQL命名完成数据操作，但这里仍然推荐使用Android提供的专用类和方法，这些类和方法的使用更加简洁、方便
- 为了使DBAdapter类支持数据添加、删除、更新和查找等功能，在DBAdapter类中增加下面的函数:
  - insert(People people)用来添加一条数据
  - queryAllData()用来获取全部数据
  - queryOneData(long id)根据id获取一条数据
  - deleteAllData()用来删除全部数据
  - deleteOneData(long id)根据id删除一条数据
  - updateOneData(long id , People people)根据id更新一条数据

SQLite数据操作

SQLite相关类

- SQLiteOpenHelper：抽象类，我们通过继承该类，然后重写数据库创建以及更新的方法， 我们还可以通过该类的对象获得数据库实例，或者关闭数据库！
- SQLiteDatabase：数据库访问类：我们可以通过该类的对象来对数据库做一些增删改查的操作
- Cursor：游标，有点类似于JDBC里的resultset，结果集！可以简单理解为指向数据库中某 一个记录的指针！

SQLiteDataBase

- SQLiteDatabase除了提供像execSQL()方法和rawQuery()方法这种直接对SQL语句解析的方法外，还针对插入、更新、删除和查询等操作专门定义了相关的方法。
  - （1）openOrCreateDatabase()方法
  - （2）execSQL()方法
  - （3）rawQuery()方法
  - （4）insert()方法
  - （5）update()方法
  - （6）delete()方法
  - （7）query()方法

8.3 数据库存储

8.3.4 数据操作

- ✓ deleteAllData()用来删除全部数据
- ✓ deleteOneData(long id)根据id删除一条数据
- ✓ updateOneData(long id , People people)根据id更新一条数据

```
1    public class DBAdapter {
2     public long insert(People people) {}
3     public long deleteAllData() { }
4     public long deleteOneData(long id) { }
5     public People[] queryAllData() {}
6     public People[] queryOneData(long id) { }
7     public long updateOneData(long id , People people){ }
8
9     private People[] ConvertToPeople(Cursor cursor){}
10   }
```

8.3 数据库存储

8.3.4 数据操作

- ConvertToPeople(Cursor cursor)是私有函数，作用是将查询结果转换为自定义的People类实例
- People类包含四个公共属性，分别为ID、Name、Age和Height，对应数据库中的四个属性值
- 重载toString()函数，主要是便于界面显示的需要
- People类的代码如下：

```
1    public class People {
2        public int ID = -1;
3        public String Name;
4        public int Age;
5        public float Height;
6
```

8.3 数据库存储

8.3.4 数据操作

```
7           @Override
8           public String toString(){
9       String result = "";
10      result += "ID：" + this.ID + "，";
11      result += "姓名：" + this.Name + "，";
12      result += "年龄：" + this.Age + "，  ";
13      result += "身高：" + this.Height + "，";
14      return result;
15          }
16      }
```

8.3 数据库存储

8.3.4 数据操作

- SQLiteDatabase类的公有函数insert()、delete()、update()和query()，封装了执行添加、删除、更新和查询功能的SQL命令
- 下面分别介绍如何使用SQLiteDatabase类的公有函数，完成数据的添加、删除、更新和查询等操作

8.3 数据库存储

8.3.4 数据操作

✓ 添加功能

- 首先构造一个ContentValues实例，然后调用ContentValues实例的put()方法，将每个属性的值写入到ContentValues实例中，最后使用SQLiteDatabase实例的insert()函数，将ContentValues实例中的数据写入到指定的数据表中
- insert()函数的返回值是新数据插入的位置，即ID值。ContentValues类是一个数据承载容器，主要用来向数据库表中添加一条数据

8.3 数据库存储

8.3.4 数据操作

- 第4行代码向ContentValues对象newValues中添加一个名称/值对，put()函数的第1个参数是名称，第2个参数是值
- 在第8行代码的insert()函数中，第1个参数是数据表的名称，第2个参数是在NULL时的替换数据，第3个参数是需要向数据库表中添加的数据

8.3 数据库存储

8.3.4 数据操作

```
1    public long insert(People people) {
2     ContentValues newValues = new ContentValues();
3
4     newValues.put(KEY_NAME, people.Name);
5     newValues.put(KEY_AGE, people.Age);
6     newValues.put(KEY_HEIGHT, people.Height);
7
8     return db.insert(DB_TABLE, null, newValues);
9    }
```

8.3 数据库存储

8.3.4 数据操作

✓ 删除功能

- 删除数据比较简单，只需要调用当前数据库实例的delete()函数，并指明表名称和删除条件即可

```
1    public long deleteAllData() {
2     return db.delete(DB_TABLE, null, null);
3    }
4
5    public long deleteOneData(long id) {
6     return db.delete(DB_TABLE,  KEY_ID + "=" + id, null);
7    }
```

8.3 数据库存储

8.3.4 数据操作

✓删除功能

- delete()函数的第1个参数是数据表名称，第2个参数是删除条件
- 在第2行代码中，删除条件为null，表示删除表中的所有数据
- 代码第6行则指明需要删除数据的id值，因此deleteOneData()函数仅删除一条数据，此时delete()函数的返回值表示被删除的数据数量

8.3 数据库存储

8.3.4 数据操作

✓ 更新功能

- 更新数据同样要使用ContentValues实例，首先构造ContentValues实例，然后调用put()函数将属性值写入到ContentValues实例中，最后使用SQLiteDatabase的update()函数，并指定数据的更新条件

```
1    public long updateOneData(long id , People people){
2     ContentValues updateValues = new ContentValues();
3     updateValues.put(KEY_NAME, people.Name);
4     updateValues.put(KEY_AGE, people.Age);
5     updateValues.put(KEY_HEIGHT, people.Height);
6
7     return db.update(DB_TABLE, updateValues,  KEY_ID + "=" + id, null);
8    }
```

8.3 数据库存储

8.3.4 数据操作

✓更新功能

- 在代码的第7行中，update()函数的第1个参数表示数据表的名称，第2个参数是更新条件。update()函数的返回值表示数据库表中被更新的数据数量

8.3 数据库存储

8.3.4 数据操作

✓ 查询功能

- 介绍查询功能前，先要介绍一下Cursor类
- 在Android系统中，数据库查询结果的返回值并不是数据集合的完整拷贝，而是返回数据集的指针，这个指针就是Cursor类
- Cursor类支持在查询结果的数据集合中以多种方式移动，并能够获取数据集合的属性名称和序号，具体的方法和说明可以参考下表

8.3 数据库存储

8.3.4 数据操作

✓ Cursor类的公有方法

- Cursor包括以下特点。
  - Cursor是每行的集合。
  - Cursor使用moveToFirst()方法定位第一行。
  - Cursor必须知道每一列的名称。
  - Cursor必须知道每一列的数据类型。
  - Cursor是一个随机的数据源。
  - 所有的数据都是通过索引取得的。

函数	说明
moveToFirst	将指针移动到第一条数据上
moveToNext	将指针移动到下一条数据上
moveToPrevious	将指针移动到上一条数据上
getCount	获取集合的数据数量
getColumnIndexOrThrow	返回指定属性名称的序号，如果属性不存在则产生异常
getColumnName	返回指定序号的属性名称
getColumnNames	返回属性名称的字符串数组
getColumnIndex	根据性名称返回序号
moveToPosition	将指针移动到指定的数据上
getPosition	返回当前指针的位置

8.3 数据库存储

8.3.4 数据操作

- ✓ 从Cursor中提取数据可以参考ConvertToPeople()函数的实现方法
- ✓ 在提取Cursor数据中的数据前，推荐测试Cursor中的数据数量，避免在数据获取中产生异常，例如下面代码的第3行到第5行
- ✓ 从Cursor中提取数据使用类型安全的get<Type>()函数，函数的参数是属性的序号，为了获取属性的序号，可以使用getColumnIndex()函数获取指定属性的序号

8.3 数据库存储

8.3.4 数据操作

```
1    private People[] ConvertToPeople(Cursor cursor){
2      int resultCounts = cursor.getCount();
3      if (resultCounts == 0 || !cursor.moveToFirst()){
4      return null;
5      }
6      People[] peoples = new People[resultCounts];
7      for (int i = 0 ; i<resultCounts; i++){
8            peoples[i] = new People();
9            peoples[i].ID = cursor.getInt(0);
10           peoples[i].Name = cursor.getString(cursor.getColumnIndex(KEY_NAME));
11           peoples[i].Age = cursor.getInt(cursor.getColumnIndex(KEY_AGE));
12           peoples[i].Height = cursor.getFloat(cursor.getColumnIndex(KEY_HEIGHT));
13           cursor.moveToNext();
14     }
15     return peoples;
16   }
```

8.3 数据库存储

8.3.4 数据操作

- 要进行数据查询就需要调用SQLiteDatabase类的query()函数，query()函数的语法如下:
- **Cursor android.database.sqlite.SQLiteDatabase.query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)**
- query()函数的参数说明

位置	类型+名称	说明
1	String table	表名称
2	String[] columns	返回的属性列名称
3	String selection	查询条件
4	String[] selectionArgs	如果在查询条件中使用通配符(?)，则需要在这里定义替换符的具体内容
5	String groupBy	分组方式
6	String having	定义组的过滤器
7	String orderBy	排序方式

8.3 数据库存储

8.3.4 数据操作

- 根据id查询数据的代码

```
1    public People[] getOneData(long id) {
2     Cursor results =  db.query(DB_TABLE, new String[] { KEY_ID, KEY_NAME,
      KEY_AGE, KEY_HEIGHT}, KEY_ID + "=" + id, null, null, null, null);
3     return ConvertToPeople(results);
4     }
```

8.3 数据库存储

8.3.4 数据操作

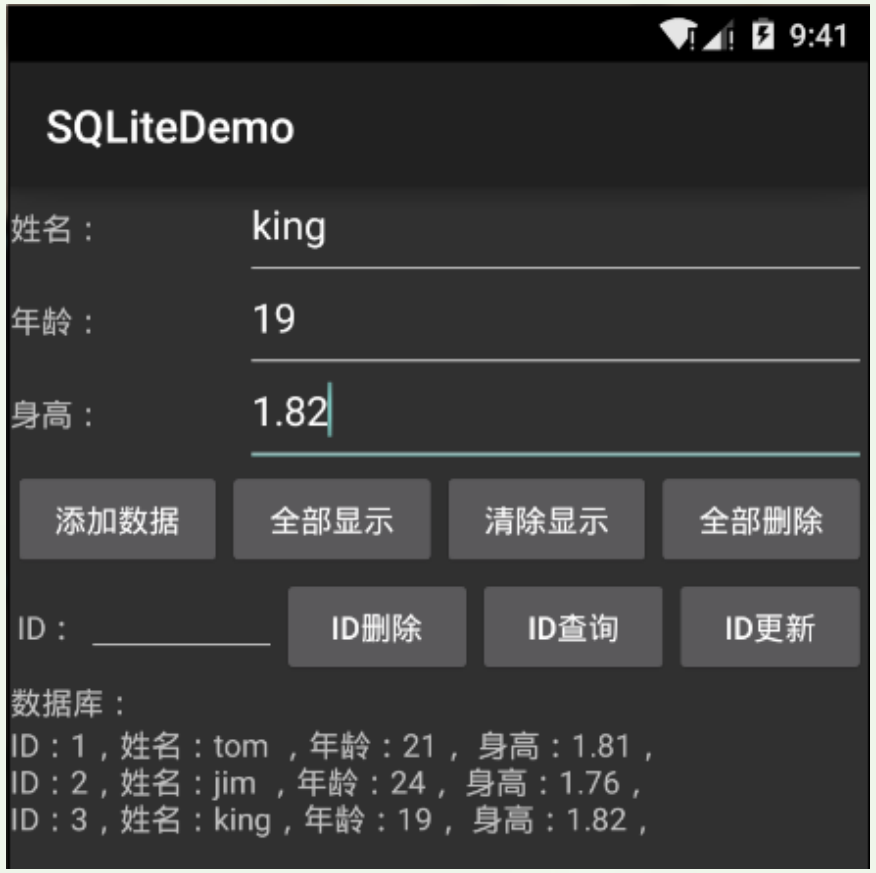
- ✓查询全部数据的代码

```
1    public People[] getAllData() {
2     Cursor results = db.query(DB_TABLE, new String[] { KEY_ID, KEY_NAME,
      KEY_AGE, KEY_HEIGHT}, null, null, null, null, null);
3     return ConvertToPeople(results);
4    }
```

8.3 数据库存储

8.3.4 数据操作

- ✓ SQLiteDemo用户界面



8.3 数据库存储

8.3.4 数据操作

- SQLiteDemo是对SQLite数据库进行操作的示例
- 在这个示例中，用户可以在界面的上方输入数据信息，通过“添加数据”按钮将数据写入数据库
- “全部显示”相当于查询数据库中的所有数据，并将数据显示在界面下方
- “清除显示”仅是清除界面下面显示数据，而不对数据库进行任何操作
- “全部删除”是数据库操作，将删除数据库中的所有数据
- 在界面中部，以“ID+功能”命名的按钮，分别是根据ID删除数据、查询数据和更新数据，而这个ID值就取自本行的EditText控件